

Path Clustering based on a Novel Dissimilarity Function for Ride-Sharing Recommenders

Eleonora D'Andrea¹, David Di Lorenzo^{2,*}, Beatrice Lazzerini¹, Francesco Marcelloni¹, Fabio Schoen³

¹Dept. of Information Engineering
University of Pisa
Largo L. Lazzarino, 56122 Pisa, Italy
eleonora.dandrea@for.unipi.it,
beatrice.lazzerini@unipi.it,
francesco.marcelloni@unipi.it

²Fleetmatics Research
Viale Mazzini 40, 50132 Florence,
Italy
david.dilorenzo@fleetmatics.com

³Dept. of Information Engineering
University of Florence
Via S. Marta 3, 50139 Florence, Italy
fabio.schoen@unifi.it

Abstract—Ride-sharing practice represents one of the possible answers to the traffic congestion problem in today's cities. In this scenario, recommenders aim to determine similarity among different paths with the aim of suggesting possible ride shares. In this paper, we propose a novel *dissimilarity function* between pairs of paths based on the construction of a *shared path*, which visits all points of the two paths by respecting the order of sequences within each of them. The shared path is computed as the shortest path on a directed acyclic graph with precedence constraints between the points of interest defined in the single paths. The dissimilarity function evaluates how much a user has to extend his/her path for covering the overall shared path. After computing the dissimilarity between any pair of paths, we execute a fuzzy relational clustering algorithm for determining groups of similar paths. Within these groups, the recommenders will choose users who can be invited to share rides. We show and discuss the results obtained by our approach on 45 paths.

Index Terms—fuzzy relational clustering, mobility patterns, path clustering, ride-sharing, smart cities.

I. INTRODUCTION

In recent years, the monitoring of city mobility has attracted growing attention due to the increasing number of vehicles (mainly private cars) causing, on the one hand, frequent traffic congestions, bottlenecks and incidents, and, on the other hand, pollution accounting for about 26% of CO₂ emissions in Europe [1]. Thus, a big effort has been recently done in the context of smart cities to monitor and reduce vehicular traffic, by improving the management of the transport networks and analyzing the dynamics of a city. This analysis focuses on: i) *traffic dynamics*, i.e., movements of vehicles in the road network, and ii) *social dynamics*, i.e., movements or grouping of people in the city, due to events and personal mobility habits. The modeling of the former allows reducing traffic congestion, addressing environmental, economic and social needs, e.g., by providing real-time information about traffic congestion and regulation, travel time estimations, incidents, pollution levels, optimal route suggestions, parking availability

[2]–[4]. The modeling of the latter allows identifying social gathering places, predicting the movements of people, estimating user similarity based on shared stay points or common paths [5]–[7]. In fact, e.g., the presence of repeated traces in the same place can indicate both a social gathering place (e.g., school, university) and a relationship between the people to whom the traces correspond (e.g., classmates).

One of the possible solutions to improve mobility in smart cities can be found in ride-sharing services. Ride-sharing is the practice according to which at least two users share a portion of a trip using the same vehicle [8]. Several car-sharing or online ride-sharing services (e.g., BlaBlaCar¹, RideshareOnline²) have spread in recent years as an economical and easy-to-use form of collaborative transportation system. Such services provide societal and environmental benefits by reducing the number of single-occupancy vehicles moving in the city. Direct economic benefits for users are money savings in fuel, tolls, parking fees. Benefits for the city are lower levels of traffic congestion and pollution. Different ride-sharing systems exist, differing from each other in some features such as the matching criterion between rides or between users, or the kind of trip (regular, commute, one-time, long-distance, short-distance, multi-hop, etc.). Thus, rides can be matched based on the Origin Destination (OD) matrix (by taking into account the origin points and/or the destination points), the pick-up and drop-off points of passengers, the keyword (cities, regions, etc.), users' needs and constraints [8]. The main challenge of ride-sharing systems is the effective recommendation, i.e., the efficient matching between rides (or users, i.e., passengers and drivers), by fulfilling the (often) conflicting objectives of meeting users' needs, respecting origin and destination points, respecting scheduling times, etc. [8]. In fact, when a user chooses a transportation mode, he/she considers several aspects of the ride: cost, travel time, flexibility, pick-up and drop-off points, privacy, etc. Some of the above-mentioned aspects are difficult to directly be controlled in public transport or long-distances ride-sharing services, where often the constraints for the pick-up/drop-off locations are not flexible for the user. On the contrary, ride-sharing allows satisfying the door-to-door

This work is partially supported by the project “Metodologie e Tecnologie per lo Sviluppo di Servizi Informatici Innovativi per le Smart Cities”, funded by “Progetti di Ricerca di Ateneo - PRA 2015” of the University of Pisa.

* Please note that the view expressed by David Di Lorenzo, from Fleetmatics, do not necessarily represent the views of the company.

¹ BlaBlaCar. www.BlaBlaCar.com

² RideshareOnline. www.RideshareOnline.com

transportation needs of users [9].

The proposed ride-sharing recommender is tailored on the urban context, where the several points of interest (POIs), along a city path represent the desired pick-up or drop-off locations for the users. We assume that users only need to be picked-up or dropped-off at the POI location. Thus, the time spent in each POI during the shared ride is negligible. In this work, we focus on studying the relationship (in terms of similarity) between users' paths, with the aim of matching similar rides by applying data mining techniques to improve e.g., car-sharing or ride-sharing services, and friend-recommendation and community-discovery systems.

Although some papers in the literature propose solutions for ride-sharing [1], [10]–[14], to the best of our knowledge, a standard method for determining the best ride-matching method does not exist [12]. In this paper, we propose an approach to group similar drivers, according to their preferred, or most frequently travelled trips, with the aim of supporting ride-sharing services [10]. Each user trip is defined in terms of a set of POIs (origin, destination, and intermediate points) forming a path in the city road network. More in detail, by considering paths pairwise, i) we define a *dissimilarity function* to determine how much two paths are close to each other; ii) we compute, according to this function, the *dissimilarity score* between pairs of paths; iii) we group the paths according to their dissimilarity score, by means of a relational clustering algorithm based on the well-known fuzzy *c*-means algorithm [15]. Clustering refers to the process of grouping a set of objects into clusters according to similarity. With respect to other works in the literature related to ride-matching, the novelty of the proposed work is the construction of a *shared path* for each pair of paths. The shared path is the shortest path, which visits all POIs of the two paths by respecting the order of sequences within each path. The shared path is found by computing the shortest path on a directed acyclic graph (DAG) representing the connections between the POIs of the two paths taken into account. Stated in other words, we aim to solve the Sequential Ordering Problem (SOP), first formulated by Escudero in [16]. SOP is a combinatorial optimization problem, which consists of determining the minimum cost Hamiltonian tour between a source node and a destination node on a directed graph, satisfying a set of precedence constraints between pairs of nodes. In this way, the user of path p_1 can be paired with the user of path p_2 in order to share with him/her the city ride at the minimal cost, in terms of additional distance to be travelled. Such a system could be very useful to reduce traffic in the city, as usual/frequent activities of citizens (e.g., grocery shopping, children pick-up and drop-off, theatre/cinema/sport events attendance) often share the same place or time. For the above reasons, it becomes easy to find people with the same transport needs. Furthermore, to enforce the order of sequences within each path is of the utmost importance, e.g., in the case of paths such as {1. *get out of work*, 2. *pick-up children at school*, 3. *go home*}, where severe ordering constraints are present.

The paper is organized as follows. Section II reviews the state of the art and the related work about dissimilarity functions and ride-sharing recommenders. Sections III and IV present the proposed system for path clustering, and the

experimental analysis, respectively. Finally, Section V provides the conclusions.

II. STATE OF THE ART

In this section, first we provide the definition of some well-known trajectory and path similarity functions, by highlighting their weaknesses and strengths. Then, we recall some ride-sharing recommenders proposed in the literature.

Similarity functions are used to evaluate the amount of similarity between objects. The class of similarity functions chosen, e.g., distance-based, statistical-based, point-based, depends on the kind of object and context considered. In fact, in the case of paths or trajectories, the function should take into account also the underlying road network graph, and problems such as graph connectivity, or compliance with the order in the sequences. A *trajectory* of a moving object is expressed as a series of discrete spatiotemporal points, taking into account both the position in the road network, and the corresponding time. A *path* instead can be considered as a spatial trajectory that covers a trip from an origin point to a destination point, and consists of a sequence of nodes and edges on a graph, representing the road network [17]. Thus, similarity functions between trajectories typically take into account position, time, speed, and direction of the object, while in the case of paths, only the spatial dimension is taken into account, i.e., the starting node, the ending node, the length of the path, the intermediate nodes, etc. The terms “trajectory” and “path” are used in this paper in an interchangeable way, even though we deal with path data mining.

The dissimilarity between objects, represented by a set of numerical attributes, is usually measured with point-based distances, e.g., L_p -norm distance metric (Euclidean distance for $p = 2$). The drawbacks of L_p -norm distance metrics are: i) the impossibility to define a distance between trajectories of different lengths, ii) the bad management of outliers, and iii) the missing management of time-shifted trajectories. Thus, L_p -norm distance metric is not well-suited for trajectories or paths. The edit distance metric [18] is defined on two strings as the minimal number of operations needed (e.g., deletion, substitution, insertion, etc.) to transform one string into the other. Several variants of the edit distance suited for paths/trajectories, have been proposed: i) the edit distance on real sequence [19] captures the similarity in shape between two trajectories; ii) the longest common subsequence (LCS) [11] is based on the matching of two sequences by stretching and rearranging the elements in time and space. The inter-cluster distance metric is used to represent the distance between paths represented as clusters of points. Examples of inter-cluster metrics are the maximum and the minimum [17]. The dynamic time warping metric [20] allows measuring the similarity between two temporal sequences, which may vary in time or speed. The drawback of this measure is its inefficiency for noisy data, its strength is the chance to be applied to trajectories of different lengths. According to the perimeter-based metric, the similarity between two paths p_1 and p_2 corresponds to the perimeter of the region formed by the two paths and the shortest path from the starting node of p_1 and the starting node of p_2 , and the shortest path from the ending node of p_1 and the ending node of p_2 [17]. This metric is well suited to compute

the similarity based on the starting and the ending nodes.

Most of the similarity functions described above are not suited to work with paths defined on a graph. Other similarity metrics handle only standalone trajectories [17], or are not able to compare paths of different length, or with different sampling frequencies. In this paper, we propose a novel dissimilarity function that tries to overcome the above-mentioned problems. The intuition behind this function is to consider as dissimilarity measure the additional road that needs to be travelled by the user of a given path to visit also the POIs of another path: if the length of this additional road is small, then the effort requested to the user for the ride sharing is negligible.

In the literature we can find several systems that can be used as ride-sharing recommenders. Ying *et al.* [21] propose a novel similarity measure between GPS trajectories, which takes into account the semantics of trajectories in order to develop a friend recommender. In [1], the authors propose a recommender capable of identifying opportunities for ride-sharing. The system collects GPS mobility data, identifies users' routine behaviors by employing text mining techniques, and finally discovers similarities among rides. He *et al.* [10] propose an intelligent routing scheme for carpooling recommendation. The system extracts frequent routes of users, searches for qualified riders and generates a commonly accepted route, which minimizes the driving distance, the walking distance, and the travel costs. Xiao *et al.* [7] estimate the similarity between users according to the semantic location histories extracted from GPS traces, with the aim of enabling friend and location recommendation. In [11], the authors tackle the ride-matching problem and perform an automatic classification of similar trajectories using the nearest neighbor classifier, and the LCS as similarity function between trajectories, and by allowing stretching in time and translating in space. In [12], the authors identify suitable matches between users based on preferred characteristic (age, gender, smoking preferences, pet restrictions, etc.) and by satisfying constraint such as, vehicle occupancy, waiting time to pick-up, number of connections, detour distance. In [14], the authors propose a dynamic ride-sharing system for taxis, by employing a shortest path algorithm and a dynamic matching criterion. Each trip is defined in terms of only the origin and destination points, and the constraints about waiting times.

Hence, the ride-matching criterion proposed in this paper is based on the geographic dissimilarities (distances) of the POIs composing the trips and not merely on the OD matrix, i.e., we take into account also trips containing intermediate stops along it. This means that paths belonging to the same region, city, district, or area of a city (as in our case) will be matched according to the distances of POIs. In addition, with the proposed ride-matching criterion, the starting/ending points of the paired paths do not have to be the same or very close in space: they can be anywhere in the city. Then, the clustering will tend to group paths based only on their real similarity.

III. THE PROPOSED SYSTEM FOR PATH CLUSTERING

In this section, we describe the proposed system for path clustering, which exploits a novel dissimilarity function between pairs of paths, and a relational fuzzy clustering algorithm based on the classical fuzzy *c*-means [15]. The

dissimilarity function evaluates how much two paths are dissimilar by exploiting a well-known algorithm for computing the shortest path on edge-weighted directed acyclic graphs (DAG). The algorithm, described in [22], is simpler and faster than the classical Dijkstra's algorithm.

The architecture of the proposed system is shown in Fig. 1. The *digital map*, which is exploited to represent the city road network and the positioning of people and vehicles in the city, is the one provided by the well-known Open Street Map (OSM)³ framework for digital maps. OSM is an open source, free-license project aimed at collecting geographic data to create freely available maps of the world with free content. A *digital map* is a graph (V, E) composed of a set of vertices V , defined as GPS positions, and by a set of edges E , each one defined in terms of length, bearing and endpoint vertices. With this structure, a road is described as the conjunction of consecutive edges (also called *segments*) identified in correspondence with intersections, changes in bearing of the road, traffic lights, pedestrian crossings, and other relevant points. The use of the digital map of the city will allow computing the travel distances between different points of the map according to the road network constraints (one-ways, limited traffic zones, etc.).

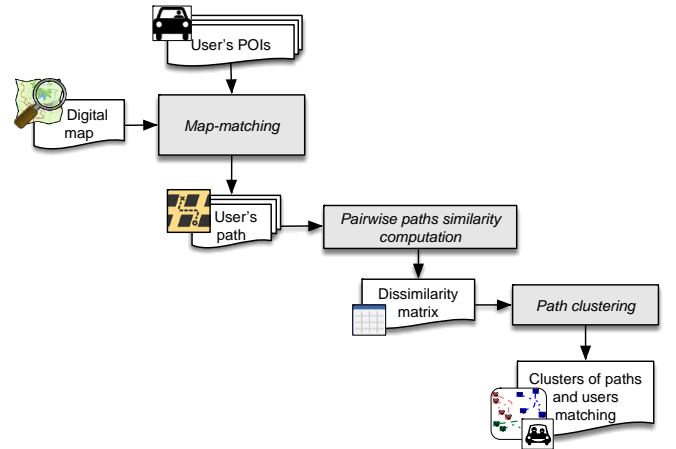


Fig. 1. The architecture of the proposed system for path clustering.

A *user POI trip* is defined in terms of a set of POIs that should be visited by the user in the given order. The set of POIs includes the path's origin and destination points, and the stops along it. These locations can be expressed with the name of the place, the complete address, or directly in terms of GPS positions (latitude and longitude). Such POIs are aligned on the road network of the digital map during the map-matching phase of the module "Map-matching" in Fig. 1. More in detail, this operation translates each POI with the corresponding GPS coordinates, and then matches the coordinates with the closest OSM map segment, by exploiting the Graph Hopper API for Java⁴. We do not discuss the details related to map-matching the POIs on the digital map, since it is not the focus of this paper. The resulting *path* is described as a sequence of GPS positions (corresponding to segments on the digital map).

³ Open Street Map. www.openstreetmap.org/

⁴ Graph Hopper Route Planner. www.graphhopper.com/

The module “*Pairwise paths similarity computation*” evaluates the similarity between pairs of paths. Given a set of paths $\Phi = \{p_1, \dots, p_P\}$, with P being the number of paths in Φ , we define two paths p_α and p_β , in Φ , as $p_\alpha = \{a_1^{(\alpha)}, \dots, a_{Q_\alpha}^{(\alpha)}\}$, and $p_\beta = \{a_1^{(\beta)}, \dots, a_{Q_\beta}^{(\beta)}\}$, with Q_α and Q_β being the number of relevant POIs in p_α and p_β , respectively. We define the *dissimilarity value* $D_{\alpha,\beta}$ between p_α and p_β as the additional length, with respect to path p_α , to be travelled by user of path p_α to visit the POIs of both paths, respecting the given orders of the POIs in paths p_α and p_β . Similarly, the *dissimilarity value* $D_{\beta,\alpha}$ between p_β and p_α corresponds to the additional length, with respect to path p_β , to be travelled by user of path p_β to visit the POIs of both paths, respecting the given orders of the POIs in paths p_β and p_α .

More formally, $D_{\alpha,\beta}$ is computed as:

$$D_{\alpha,\beta}(p_\alpha, p_\beta) = \text{length}(p_\alpha \cup p_\beta) - \text{length}(p_\alpha) \quad (1)$$

where $p_\alpha \cup p_\beta$ is the shortest *shared path* travelled by the user of path p_α for visiting each point of paths p_α and p_β , preserving the order of the visited POIs in p_α and p_β , $\text{length}(p_\alpha \cup p_\beta)$ is the length of the *shared path* $p_\alpha \cup p_\beta$, and $\text{length}(p_\alpha)$ is the length of path p_α .

$\text{Length}(p_\alpha)$ is computed as the sum of the distances between consecutive points in p_α :

$$\text{length}(p_\alpha) = \sum_{k=1}^{Q_\alpha-1} \text{dist}(a_k^{(\alpha)}, a_{k+1}^{(\alpha)}) \quad (2)$$

where $a_k^{(\alpha)}$ and $a_{k+1}^{(\alpha)}$ are two consecutive POIs of path p_α , and $\text{dist}(a_k^{(\alpha)}, a_{k+1}^{(\alpha)})$ is the length of the shortest path between $a_k^{(\alpha)}$ and $a_{k+1}^{(\alpha)}$ computed on the map by using the Graph Hopper API. Thus, $\text{dist}(a_k^{(\alpha)}, a_{k+1}^{(\alpha)})$ corresponds to the real travel distance between $a_k^{(\alpha)}$ and $a_{k+1}^{(\alpha)}$ on the city road network. The dissimilarity function in (1) respects the coincidence axiom, that is, $D_{\alpha,\beta}(p_\alpha, p_\beta) = 0$ if and only if $p_\alpha = p_\beta$. On the contrary, the symmetry property is not satisfied, that is, $D_{\alpha,\beta} \neq D_{\beta,\alpha}$.

The computation of $\text{length}(p_\alpha \cup p_\beta)$ is not trivial since we have to determine the shortest path travelled by the user of path p_α for visiting each point of the paths p_α and p_β , preserving the order of the visited POIs in p_α and p_β . To this aim, we build an edge-weighted DAG. The graph has $N = Q_\alpha \cdot (Q_\beta + 1) + (Q_\alpha + 1) \cdot Q_\beta + 1$ nodes defined as follows:

- $Q_\alpha \cdot (Q_\beta + 1)$ nodes of type $X_{i,j}$, with $i = 1, \dots, Q_\alpha$, and $j = 0, \dots, Q_\beta$;
- $(Q_\alpha + 1) \cdot Q_\beta$ nodes of type $Y_{i,j}$ with $i = 0, \dots, Q_\alpha$, and $j = 1, \dots, Q_\beta$;
- 1 destination node D corresponding to the end of the *shared path*.

The nodes of type $X_{i,j}$ and $Y_{i,j}$ are associated with, respectively, the POIs $a_i^{(\alpha)}$ of path p_α and the POIs $a_j^{(\beta)}$ of path p_β and indicate the progress achieved in completing the *shared*

path. More in detail, staying in node $X_{i,j}$ of the DAG means that the *shared path* has just visited the POI $a_i^{(\alpha)}$ of path p_α and has already visited all the previous POIs $a_1^{(\alpha)}, \dots, a_{i-1}^{(\alpha)}$ of path p_α and the POIs $a_1^{(\beta)}, \dots, a_j^{(\beta)}$ of path p_β . Similarly, staying in node $Y_{i,j}$ means that the *shared path* has just visited the POI $a_j^{(\beta)}$ of path p_β and has already visited all the previous POIs $a_1^{(\beta)}, \dots, a_{j-1}^{(\beta)}$ of path p_β and the POIs $a_1^{(\alpha)}, \dots, a_i^{(\alpha)}$ of path p_α .

The graph contains $N + 2$ edges. The weight associated with each edge is the minimal distance computed on the road network between the considered POIs. N edges are built according to the following rules:

- $X_{i,j} \rightarrow X_{i+1,j}$, with weight $\text{dist}(a_i^{(\alpha)}, a_{i+1}^{(\alpha)})$
- $X_{i,j} \rightarrow Y_{i,j+1}$, with weight $\text{dist}(a_i^{(\alpha)}, a_{j+1}^{(\beta)})$
- $Y_{i,j} \rightarrow X_{i+1,j}$, with weight $\text{dist}(a_j^{(\beta)}, a_{i+1}^{(\alpha)})$
- $Y_{i,j} \rightarrow Y_{i,j+1}$, with weight $\text{dist}(a_j^{(\beta)}, a_{j+1}^{(\beta)})$

The last 2 edges connect the nodes X_{Q_α, Q_β} and Y_{Q_α, Q_β} with the destination node D and have a weight equal to 0.

The *shared path* is computed as the shortest path on the DAG using the code included in the library `algs4`⁵ for Java. For each pair of paths p_α and p_β , two *shared paths* are computed. The first one ($p_\alpha \cup p_\beta$) starts from point $a_1^{(\alpha)}$ (corresponding to node $X_{1,0}$ in the DAG) and is travelled by user of path p_α ; the second one ($p_\beta \cup p_\alpha$) starts from $a_1^{(\beta)}$ (corresponding to node $Y_{0,1}$ in the DAG) and is travelled by user of path p_β . Both the *shared paths* end in the destination node D of graph DAG.

Fig. 2 shows an example with two paths: p_α with $Q_\alpha = 3$, and p_β with $Q_\beta = 2$. The user of path p_α leaves home, drops-off children at school, and finally goes to work; the user of path p_β , e.g., a teacher of the same school, leaves home and reaches the school. Fig. 3 shows the corresponding DAG built for the pair of paths. The DAG has 18 nodes and 20 edges. We compute the dissimilarity value for each pair of paths in Φ . We obtain a $P \times P$ dissimilarity matrix of values $D_{\alpha,\beta}$ computed according to (1), and representing the mutual relationship between pairs of paths.

Then, in module “*Path clustering*”, similar paths are grouped by means of a relational clustering algorithm, according to the pairwise dissimilarity value. We employ as relational clustering the ARCA (Any Relational Clustering Algorithm) algorithm proposed in [15]. This algorithm considers each path p_α as an object described by P features, where each feature β corresponds to the value of dissimilarity between p_α and p_β . Thus, the relational clustering problem is transformed into an object clustering problem, which is tackled by adopting the well-known fuzzy c -means clustering algorithm [23]. ARCA partitions the dataset by minimizing the Euclidean distance between each path (described by the P dissimilarity values) belonging to a cluster and the prototype of the cluster. ARCA has proved to be more stable and effective

⁵ <http://algs4.cs.princeton.edu/code/>

than popular fuzzy relational clustering algorithms proposed in the literature. In this paper, we adopt the cosine distance in place of the Euclidean distance for coping with the high-dimensionality of the objects. The cosine distance is computed as the angle between the feature (dissimilarity value) vectors representing the objects (in our case, the paths).

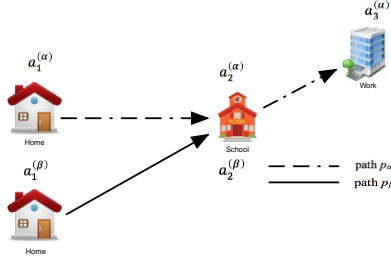


Fig. 2. An example of two paths (p_α : Home-School-Work, and p_β : Home-School).

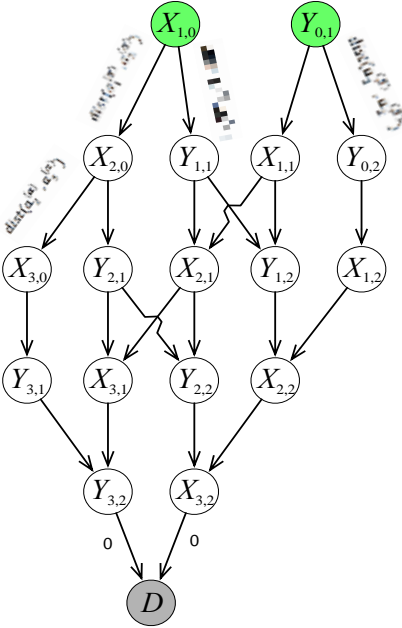


Fig. 3. The corresponding DAG built for the pair of paths in Fig. 2. Please note that, for the sake of simplicity, we show the weights only on four edges, but all the weights are computed as explained in the text.

IV. EXPERIMENTAL ANALYSIS

In this section we describe the case study adopted to evaluate the proposed system. We first generated 45 paths in different areas of the city of Pisa. Then, we computed the dissimilarity between these paths and applied the ARCA clustering algorithm. Finally, we analyzed the clusters obtained.

A. Path Generation

The 45 paths used in the experimental analysis were randomly generated in the road network of the city of Pisa, Italy (corresponding to an area of about 70 km²). The paths belong to about 3 main areas of the city and have a length ranging from 150 m to 3 km (we recall that we focus on urban ride-sharing, thus the length of the paths may be shorter with respect to traditional inter-city ride-sharing systems). For the

aim of this paper, they represent an example of the most frequent or usual paths of users moving in the city by car, in the time interval of about 1 hour. In fact, the aim is to analyze trips with similar temporal constraints, i.e., users with similar needs, and moving approximately in a similar area. Each path consists, e.g., of a set of Q POIs (e.g., school, grocery, department store, drugstore) or other relevant points (e.g., user home, user work place) for the user of the path. The value of Q ranges from 2 (meaning that the path is defined only in terms of its origin and destination points) to 5. Each POI can be provided as the name of the place, the complete address, or directly by means of its GPS coordinates. The POIs are then map-matched and translated in GPS coordinates.

B. Clustering of Paths

We computed the elements of the $P \times P$ dissimilarity matrix and applied the ARCA algorithm. As usual, we set the fuzzification coefficient of the fuzzy c -means to 2. We executed the ARCA algorithm with $c = 2$, $c = 3$, and $c = 4$ clusters. We employed a well-known cluster validity index, i.e., the Xie-Beni index [24], to determine the optimal number of clusters. In fact, a typical practice suggested in [25] is to consider as the optimal number of clusters the first minimum of the Xie-Beni index. The first minimum for the Xie-Beni index was found for $c = 3$ clusters, as shown in Fig. 4. Paths having high membership degree (larger than $\omega = 0.5$) to one cluster (say cluster v), and low membership degrees to the remaining clusters are associated with cluster v .

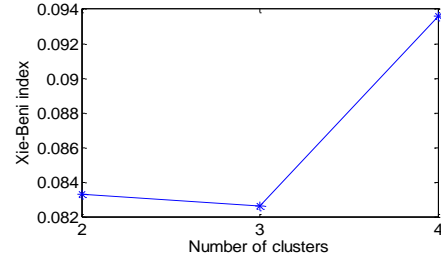


Fig. 4. The Xie-Beni index values.

Table I shows the results (in terms of membership degrees) of the execution of the ARCA clustering algorithm, with the number $c = 3$ of clusters identified by the Xie-Beni index. Figs. 5-7 show, respectively, the 3 clusters of paths on the city map. The respective positioning of the 3 areas of the city, i.e., the clusters to which approximately the paths belong, can be easily determined thanks to the star symbol, indicating the city center. Please note that some paths close to the boundary of the figures may start/end out of the figure. As the table shows, the higher membership degree is always greater than 0.75 for all the paths except for 3 paths, namely paths #20, #23, and #24, where all the membership values are lower than ω . These three paths were assigned to no cluster. By observing the paths on the map (Fig. 6 for path #23 and Fig. 7 for paths #20 and #24), we can note that paths #20, #23 and #24 are very short paths i.e., they have a length lower than 200 m and are quite far from the other paths.

The results show that the *Path clustering* module successfully performs the clustering of paths. In fact, as done in other papers [26], to obtain the “ground truth” clustering

results, we manually labelled the paths as belonging to 3 areas of the city, and then we checked the labelling with the clustering performed by the system.

TABLE I. MEMBERSHIP DEGREES OF EACH PATH TO THE THREE CLUSTERS.

#path	Membership degree to #cluster 1	Membership degree to #cluster 2	Membership degree to #cluster 3	#cluster
1	0.99148994	0.00244746	0.00606259	1
2	0.95490033	0.00812565	0.03697400	1
3	0.99559369	0.00150113	0.00290516	1
4	0.99201985	0.00274745	0.00523269	1
5	0.98793974	0.00374690	0.00831334	1
6	0.00995498	0.01044541	0.97959959	3
7	0.00914573	0.00626595	0.98458831	3
8	0.02743599	0.00504299	0.96752101	3
9	0.00965743	0.01033874	0.98000382	3
10	0.01611162	0.00338497	0.98050339	3
11	0.07048132	0.76873916	0.16077950	2
12	0.02730657	0.03161453	0.94107889	3
13	0.06333103	0.78077246	0.15589650	2
14	0.06443096	0.78260220	0.15296682	2
15	0.06288524	0.77353835	0.16357640	2
16	0.94718076	0.00764978	0.04516944	1
17	0.77291616	0.02740858	0.19967525	1
18	0.99148761	0.00270049	0.00581188	1
19	0.97500981	0.00632495	0.01866522	1
20	0.48950549	0.05669654	0.45379795	-
21	0.01002262	0.97290460	0.01707276	2
22	0.01366032	0.96555733	0.02078234	2
23	0.43597302	0.45759169	0.10643527	-
24	0.10679680	0.42155047	0.47165272	-
25	0.13720656	0.01773537	0.84505806	3
26	0.00822758	0.97995014	0.01182227	2
27	0.00824899	0.98008631	0.01166469	2
28	0.01119239	0.97208018	0.01672742	2
29	0.00303132	0.99229453	0.00467413	2
30	0.11954798	0.81440203	0.06604997	2
31	0.86424369	0.03422620	0.10153010	1
32	0.02184706	0.01402594	0.96412698	3
33	0.01261069	0.97105932	0.01632998	2
34	0.00509321	0.98744987	0.00745690	2
35	0.05099951	0.08574314	0.86325734	3
36	0.97332568	0.00725452	0.01941979	1
37	0.05350623	0.87537348	0.07112027	2
38	0.01251605	0.96792623	0.01955771	2
39	0.97596058	0.01075760	0.01328180	1
40	0.06791020	0.76752092	0.16456887	2
41	0.00582395	0.98424635	0.00992969	2
42	0.01182435	0.96959419	0.01858145	2
43	0.88935471	0.03789428	0.07275099	1
44	0.00195688	0.99508776	0.00295535	2
45	0.00656585	0.98616225	0.00727189	2

C. Recommendation of rides

Once the clusters have been determined, we can exploit them for a ride-sharing recommender. Indeed, paths belonging to the same cluster are likely to be quite close to each other. This means, for instance, that a driver can share her/his car with other users who travel along paths belonging to the same cluster, with acceptable deviations. The recommender can try to match the users registered to the service starting from the least dissimilar paths in the cluster. In addition, the proposed matching criterion, since it is based on the clustering of paths,

provides *groups*, not simply *pairs*, of similar, i.e., near, paths, by increasing the possibilities for the users to share a ride. Obviously, it is always possible to select the best matching pair of paths in a group by checking the lowest dissimilarity score value.

V. CONCLUSIONS

We have presented a system for clustering similar paths in a city road network, with the aim of supporting ride-sharing recommendation. Users of paths belonging to the same cluster can be likely interested in ride-sharing. The ride-matching criterion is based on the values of a novel dissimilarity function between pairs of paths. The *dissimilarity function* computed on two paths p_α and p_β measures the length of the minimal additional distance to be travelled by the user of path p_α to visit all POIs of path p_β , by respecting the precedence constraints between the points of each path. The problem can be formulated and solved as a Sequential Ordering Problem, aimed to find the shortest path that satisfies the precedence constraints. Then, a fuzzy relational clustering algorithm is employed for determining groups of similar paths, based on the dissimilarity values between paths. We have discussed the application of our system to 45 paths in the city of Pisa and have shown that it is able to cluster these paths successfully.

As future work, we are going to implement the developed system on a cloud computing architecture, with the aim of guaranteeing elasticity and scalability. Further, we are planning to evaluate the user satisfaction by explicit feedbacks from the users of the service.

REFERENCES

- [1] N. Bicocchi and M. Mamei, "Investigating ride sharing opportunities through mobility data analysis", *Pervasive and Mobile Comp.*, vol. 14, pp. 83–94, 2014.
- [2] N. Pelekis, I. Kopanakis, M. Gerasimos, I. Ntoutsis, G. Andrienko and Y. Theodoridis, "Similarity search in trajectory databases", in *Proc. Int. Symp. Temporal Representation and Reasoning (TIME)*, Alicante, Spain, 2007, pp. 129–140.
- [3] G. Anastasi, M. Antonelli, A. Bechini, S. Brienza, E. D'Andrea, D. De Guglielmo, P. Ducange, B. Lazzarini, F. Marcelloni and A. Segatori, "Urban and social sensing for sustainable mobility in smart cities", in *Proc. IFIP/IEEE Int. Conf. Sust. Internet and ICT for Sustain. (SustainIT)*, Palermo, Italy, 2013, pp. 1–4.
- [4] E. D'Andrea, P. Ducange, B. Lazzarini and F. Marcelloni, "Real-Time Detection of Traffic From Twitter Stream Analysis", *IEEE Trans. Intell. Transp. Syst.*, vol. 16, no. 4, pp. 2269–2283, 2015.
- [5] Y. Zheng, L. Zhang, X. Xie and W.-Y. Ma, "Mining interesting locations and travel sequences from GPS trajectories", in *Proc. 18th Int. Conf. World Wide Web (WWW)*, Madrid, Spain, 2009, pp. 791–800.
- [6] T.M.T. Do and D. Gatica-Perez, "Contextual conditional models or smartphone-based human mobility prediction", in *Proc. 2012 ACM Conf. Ubiquitous Comp. (UbiComp)*, Pittsburgh, Pennsylvania, 2012, pp. 163–172.
- [7] X. Xiao, Y. Zheng, Q. Luo and X. Xie, "Finding similar users using category-based location history", in *Proc. 18th SIGSPATIAL Int. Conf. Advances Geo. Inf. Syst. (GIS)*, San Jose, CA, 2010, pp. 442–445.
- [8] M. Furuhashi, M. Dessouky, F. Ordóñez, M.-E. Brunet, X. Wang

- and S. Koenig, “Ridesharing: The state-of-the-art and future directions”, *Transp. Research Part B: Methodological*, vol. 57, pp. 28–46, 2013.
- [9] M. Piórkowski, “Collaborative transportation systems”, in *Proc. 2010 IEEE Wireless Comm. and Networking Conf. (WCNC)*, Sydney, Australia, 2010, pp. 1–6.
- [10] W. He, K. Hwang and D. Li, “Intelligent Carpool Routing for Urban Ridesharing by Mining GPS Trajectories”, *IEEE Trans. on Intelligent Transp. Syst.*, vol. 15, no. 5, pp. 2286–2296, 2014.
- [11] M. Vlachos, G. Kollios and D. Gunopulos, “Discovering similar multidimensional trajectories”, in *Proc. 18th Int. Conf. Data Eng. (ICDE)*, San Jose, CA, 2002, pp. 673–684.
- [12] D. Teodorović and M. Dell’Orco, “Mitigating Traffic Congestion: Solving the Ride-Matching Problem by Bee Colony Optimization”, *Transp. Planning and Technology*, vol. 31, no. 2, pp. 135–152, 2008.
- [13] K. Ghoseiri, A. Haghani, and M. Hamedi (2011, January), Real-time rideshare matching problem. Final Report UMD-2009-05. Mid-Atlantic Universities Transportation Center, US.
- [14] C. Tian, Y. Huang, Z. Liu, F. Bastani, and R. Jin, “Noah: a dynamic ridesharing system”, in *Proc. 2013 ACM SIGMOD Int. Conf. Manage. Data (SIGMOD)*, New York, NY, 2013, pp. 985–988.
- [15] P. Corsini, B. Lazzerini and F. Marcelloni, “A new fuzzy relational clustering algorithm based on the fuzzy C-means algorithm”, *Soft Computing*, vol. 9, no. 6, pp. 439–447, 2005.
- [16] L.F. Escudero, “An inexact algorithm for the sequential ordering problem”, *European J. of Operational Research*, vol. 37, no. 2, pp. 236–249, 1988.
- [17] Q. Lu, F. Chen and K. Hancock, “On path anomaly detection in a large transportation network”, *Computers, Env. and Urban Syst.*, vol. 33, no. 6, pp. 448–462, 2009.
- [18] V. Levenshtein, “Binary codes capable of correcting deletions, insertions, and reversals”, *Soviet Physics—Doklady* vol. 10, no. 10, pp. 707–710, 1966.
- [19] L. Chen, M. T. Özsü and V. Oria, “Robust and fast similarity search for moving object trajectories”, in *Proc. 2005 ACM SIGMOD Int. Conf. Manage. of Data (SIGMOD)*, Baltimore, MD, 2005, pp. 491–502.
- [20] D. Berndt and J. Clifford, “Using dynamic time warping to find patterns in time series”, in *Proc. AAAI ‘94 Workshop Knowl. Discovery in Databases*, Seattle, WA, 1994, pp. 229–248.
- [21] J.J.-C. Ying, E.H.-C. Lu, W.-C. Lee, T.-C. Weng and V.S. Tseng, “Mining user similarity from semantic trajectories”, in *Proc. 2nd SIGSPATIAL Int. Workshop Location Based Social Networks (LBSN)*, San Jose, CA, 2010, pp. 19–26.
- [22] R. Sedgewick and K. Wayne, *Algorithms, 4th Edition*. Addison-Wesley Professional, Boston, MA, 2011.
- [23] J.C. Bezdek, *Pattern Recognition With Fuzzy Objective Function Algorithms*. Springer-Verlag US, 1981.
- [24] X.L. Xie and G. Beni, “A validity measure for fuzzy clustering”, *IEEE Trans. Pattern Anal., Mach. Intell.*, vol. 13, no. 8, pp. 841–847, 1991.
- [25] M. Setnes and R. Babuška, “Fuzzy relational classifier trained by fuzzy clustering”, *IEEE Trans. Syst., Man, and Cybernetics, Part B: Cybernetics*, vol. 29, no. 5, pp. 619–625, 1999.
- [26] G.-P. Roh and S.-W. Hwang, “Nncluster: An efficient clustering algorithm for road network trajectories”, in *Proc. 15th Int. Conf. Database Systems Adv. App. (DASFAA)*, Tsukuba, Japan, 2010, pp. 47–61.

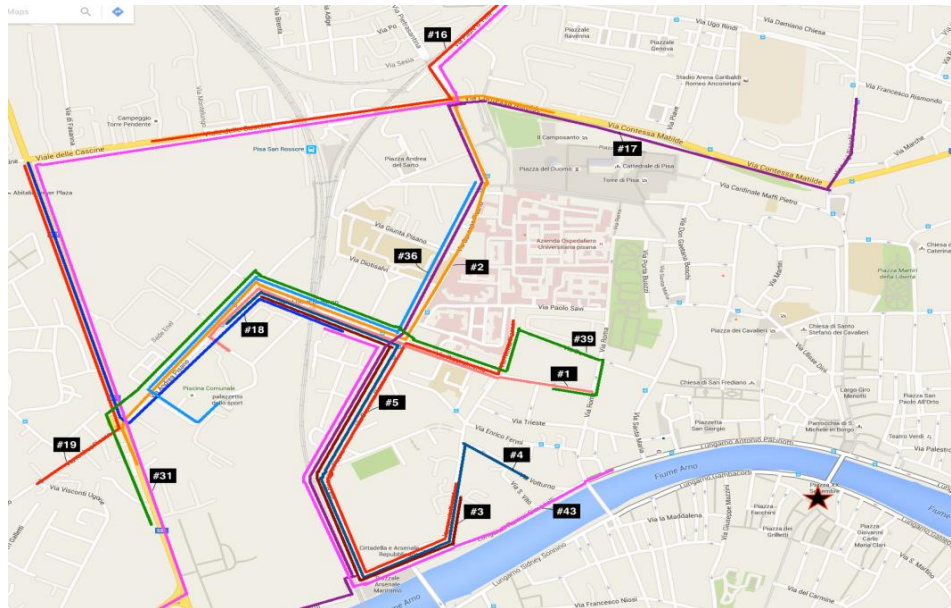


Fig. 5. Paths corresponding to cluster #1. (The underlying map is provided by Google®).

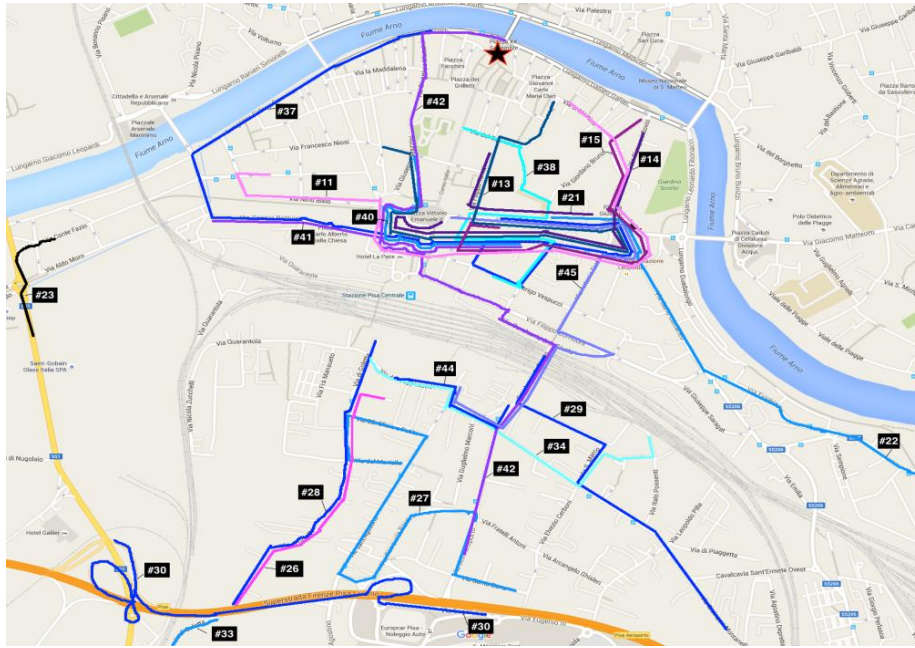


Fig. 6. Paths corresponding to cluster #2 and path #23. (The underlying map is provided by Google®).

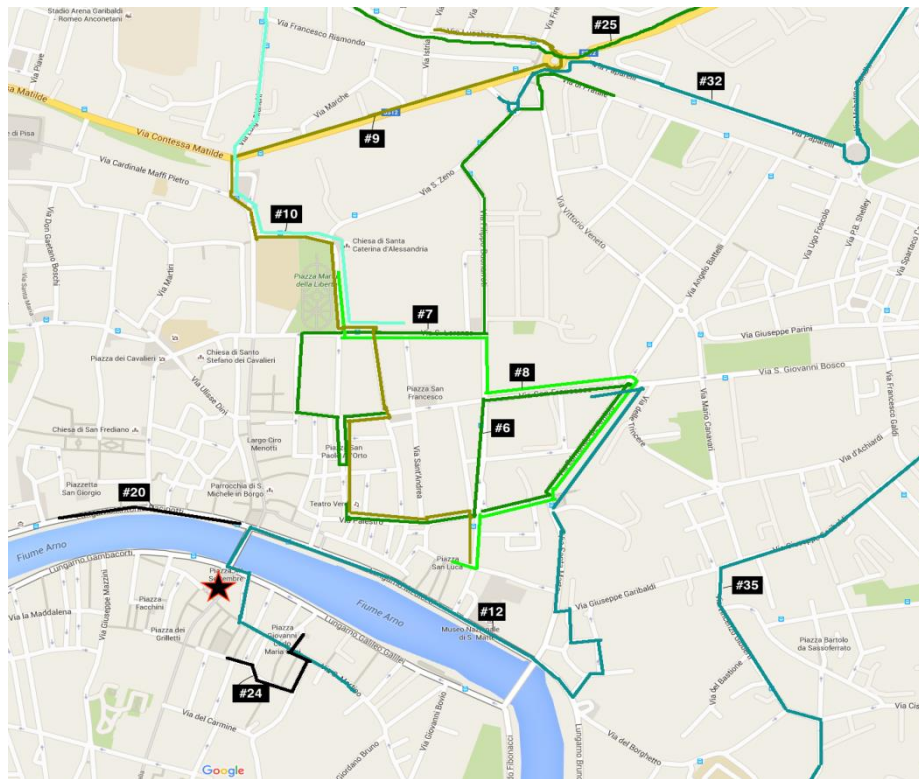


Fig. 7. Paths corresponding to cluster #3 and paths #20 and #24 (The underlying map is provided by Google®).